Automatic Program Rewriting for Non-Ground Answer Set Programs

Nick Hippen & Yuliya Lierler

University of Nebraska at Omaha

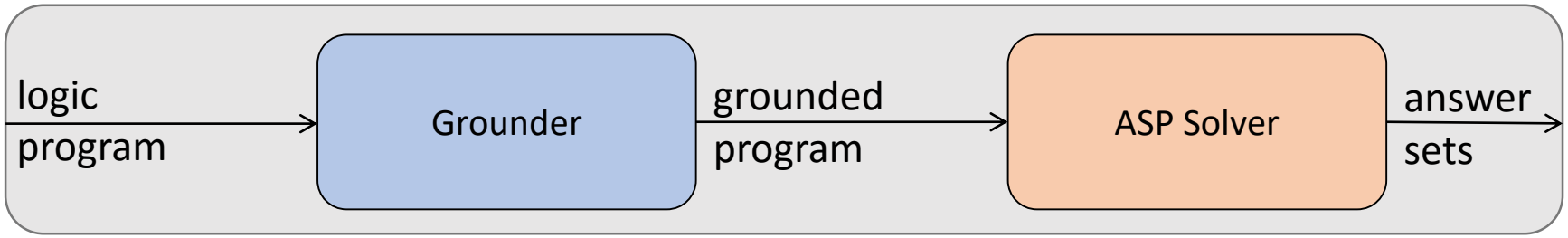
BACKGROUND

What is Answer Set Programming (ASP)?

- Constraint programming paradigm geared towards solving difficult combinatorial search problems
- Prolog-like syntax

Logic Rule	Meaning
$child(X,Y) \leftarrow parent(Y,X).$	*X is a child of Y if Y is a parent of X.*
$innocent(X) \leftarrow not\ guilty(X).$ Head ← Body	*X is innocent if I have no reason to believe that X is guilty*

ASP Solver Architecture & Grounding



Logic Program	Grounded Program	Intelligently Grounded Program
$parent(bob, ally).$ $parent(mary, john).$ $sibling(bob, mary).$ $cousin(X, Y) \leftarrow parent(P1, X),$ $\quad parent(P2, Y),$ $\quad sibling(P1, P2),$ $\quad X \neq Y.$	$parent(bob, ally).$ $parent(mary, john).$ $sibling(bob, mary).$ $cousin(john, mary) \leftarrow parent(ally, john),$ $\quad parent(bob, mary),$ $\quad sibling(ally, bob),$ $\quad john \neq mary.$ … $cousin(bob, bob) \leftarrow parent(bob, bob),$ $\quad parent(bob, bob),$ $\quad sibling(bob, bob),$ $\quad bob \neq bob.$ …	$parent(bob, ally).$ $parent(mary, john).$ $sibling(bob, mary).$ $cousin(ally, john) \leftarrow parent(bob, ally),$ $\quad parent(mary, john),$ $\quad sibling(bob, mary),$ $\quad ally \neq john.$

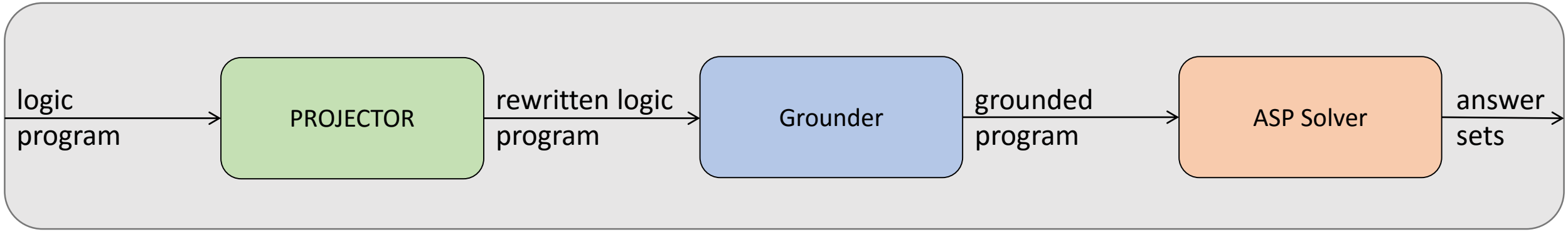
Motivations

- Intuitive encodings are not always the most optimal
- Fine-tuning requires expertise
  - Smaller grounding sizes often translate into faster solve times
  - Eliminating variables from rules can lead to smaller grounding size

Question

- What fine-tuning can be done automatically via program rewriting techniques?

ASP Solver Architecture Extended



Our Direction

- Inspired by relational databases
- Two considered rewritings
  - α-projection
  - β-projection

Our Direction

- Inspired by relational databases
- Two considered rewritings
  - α-projection
  - **β-projection**

β-PROJECTION

β-Projection

Goal: Eliminate any unnecessary variables from the rule by isolating them in a newly introduced rule.

- Need to make sure this rule does not contain the same variables as the original
- Carry “guards” for variables when possible
  - Attempts to restrict the domain of variables in a rule as much as possible without introducing new variables

Example: PermutationPatternMatching

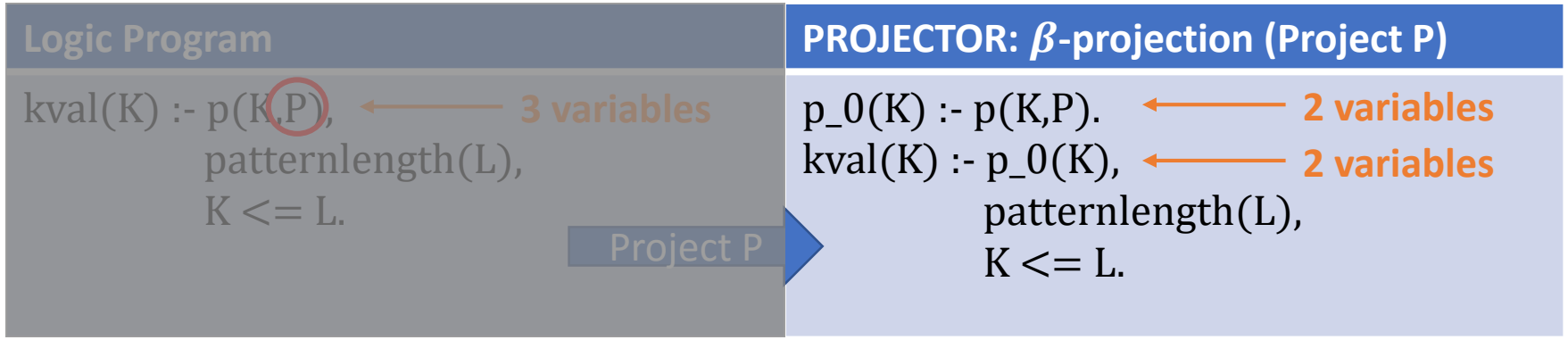
Logic Program

```
kval(K) :- p(K,P),
           patternlength(L),
           K <= L.
```

Consider if K,P,L have 100 possible distinct ground values for this rule.

$$TotalGrounding_{orig} = |K| * |P| * |L| = 100^3 = 1{,}000{,}000$$

Example: PermutationPatternMatching



Consider if K,P,L have 100 possible distinct ground values for both rules.

$$GroundingSize_{p\_0} = |K| * |P| = 100^2 = 10{,}000$$
$$GroundingSize_{kval} = |K| * |L| = 100^2 = 10{,}000$$
$$TotalGrounding_{proj} = GroundingSize_{p\_0} + GroundingSize_{kval} = 20{,}000$$

More Details… But No Time…

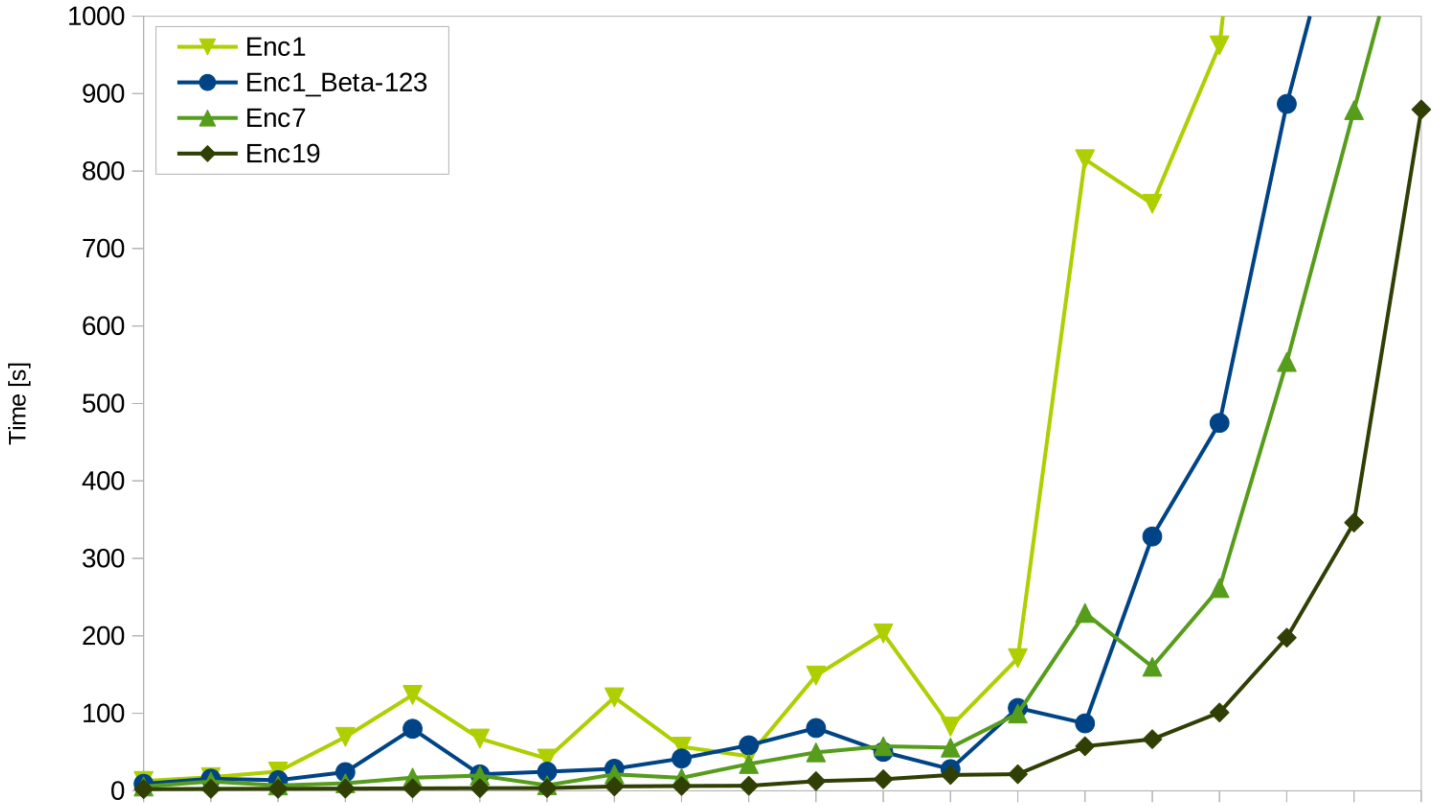
- Supported languages features
- Variable selection
  - Non-deterministic behavior
- Safety
- Algorithm & Implementation Details

EXPERIMENTAL RESULTS

Experimental Analysis

- **ASPCCG**: ASP based natural language parser
  - 3 encodings of increasing levels of human optimization
    - Created by Matthew Buddenhagen, Yuliya Lierler & Peter Schuller
  - Enc1: No human optimization
  - Enc7: Moderate human optimization
  - Enc19: Notable human optimization
- **ASPComp14**
  - Stable Marriage
  - Permutation Pattern Matching
  - Knight Tour with Holes
  - Misc.

ASPCCG: Overall



Related Work: Lpopt

- Lpopt (Bichler, Morak, Woltran, 2016)
  - Very similar ideas (reduce # of variables in rules)
  - Different process

Bichler, Manuel. "Optimizing non-ground answer set programs via rule decomposition." Bachelor thesis. TU Wien (2015).

Bichler, Manuel, Michael Morak, and Stefan Woltran. "lpopt: A rule optimization tool for answer set programming." *International Symposium on Logic-Based Program Synthesis and Transformation*. Springer, Cham, 2016.

Full ASPComp 2014 Benchmarks

- 22 total benchmarks
  - 4 benchmarks had no rewrites from both lpopt & projector
  - 6 benchmarks had rewrites but lpopt & projector made no significant performance effect
  - 12 benchmarks had rewrites with performance impact
- 20 instances for each program
- Timeout of 5 minutes

Lpopt Success

- Lpopt performed better on average than Projector on 3 programs
  - Hanoi Tower
    - Projector performed worse than base encoding
  - Labyrinth
    - Lpopt solved 19 of 20 instances vs. 18 of 20 for base encoding & 17 of 20 for Projector
  - Nomistery
    - Projector performed better than base encoding but ~50% worse than Lpopt

Projector Success

- Lpopt performed better on average than both the base encoding & projector on 3 programs
  - Graph Colouring
  - Knight Tour With Holes
  - Permutation Pattern Matching
    - Significant improvement

Similar Rewrite Performance

- Both Lpopt & Projector hurt performance:
  - Minimal Diagnosis
  - Valves Location Problem
- Both Lpopt & Projector improve performance:
  - Ricochet Robots
- No significant difference between Lpopt & Projector in both cases

Conclusions, Current & Future Work

Conclusions

- Automatic rewriting techniques are worth exploring
- System PROJECTOR available on the UNO NLPKR Lab website
- "Automatic Program Rewriting in Non-Ground Answer Set Programs" (Hippen, Lierler, 2019) presented at *International Symposium on Practical Aspects of Declarative Languages 2019 (PADL 2019)*

Current Work

- Improve language support

Future Work

- Better heuristics for variable selection (grounding size prediction)

Acknowledgements

University of Nebraska Omaha

- Brian Hodges
- Jacob Lee

University of Kentucky

- Michael Dingess
- Daniel Houston
- Liu Liu
- Shelby Stocker
- Dr. Mirek Truszczynski

Misc.

- Roland Kaminski
- Stefan Woltran

NSF Project: Automated Optimization of Programs and Processing Tools in Answer Set Programming Optimization

Thank You!

Questions?